

LAIRD LINUX SOFTWARE INTEGRATION GUIDE

Application Note

v. 1.1

NOTES FOR FIRST INTEGRATION

This document explains the steps required to fully integrate the SD45 and MSD45 Linux software package.

We recommend that you thoroughly analyse each step of the process. Each individual step should be integrated separately and manually tested. Attempting all of the steps at once without testing, will likely cause bugs that are difficult to troubleshoot and confusing to untangle.

Integrating a Wi-Fi driver into specific Linux platform code can be challenging and may require platform-specific changes. If the following information is confusing or does not provide the proper results, please check <https://laird-ews-support.desk.com/> for further assistance.

INTEGRATION

Ready Your Board Support Package

Before integrating the Laird Linux software package, disable all device software from attempting to manage the wireless connection. This includes, but is not limited to, NetworkManager, Connman, wpa_supplicant, and others. Laird also expects that the customer's platform supplies the DHCP solution for use with the wireless interface. Laird can provide guidance if needed on this subject. The WB45 software release can be used as a guide for implementing a DHCP solution if your board support package doesn't already include one. Contact Laird Support for more details.

Supporting Libraries and Programs

Before integration of Laird's enhanced wireless software package, several prerequisite libraries and programs should be added to your device's root file system. We recommend that they be added through your device's board support package if possible.

- **libnl** – libnl is the set of libraries used by userspace programs to communicate to the kernel's netlink interface. Laird recommends using version 3.2.25 or greater if possible. For more information visit <http://www.infradead.org/~tgr/libnl/>
- **CRDA** – CRDA is the userspace daemon that is responsible for managing the Linux kernel's wireless regulatory database. If your board support package doesn't have support for CRDA, see the section on building a regulatory database into the cfg80211 kernel module in the Laird Backports Driver section. For more information on CRDA see <https://wireless.wiki.kernel.org/en/developers/regulatory/crda>
- **wireless-regdb 2013.02.13** – Laird recommends a specific regulatory database to satisfy Linux kernel requirements for one to exist. If the wireless-regdb 2013.02.13 can't be installed from your board support package, download the package from <https://www.kernel.org/pub/software/network/wireless-regdb/wireless-regdb-2013.02.13.tar.bz2> and extract the archive. If using CRDA, find the regulatory.bin and add it to your device's root filesystem at /usr/lib/crda/. Otherwise, keep the db.txt file for use in the Laird Backports Driver. For more information visit <https://wireless.wiki.kernel.org/en/developers/regulatory/wireless-regdb>

Laird Backports Driver

Backports is a software package that allows Linux users to port a driver from a newer Linux kernel back to an older Linux kernel. Laird has created a version of Linux Backports that contains all Laird customizations to the ath6kl driver provided with the Linux kernel. The current Laird Backports releases are based on the 3.13 kernel and generate ath6kl drivers for kernel versions 2.6.26 to 3.13.

To download the release, visit the Laird GitHub site: <https://github.com/LairdCP/laird-linux-backports/releases>

Download the Laird Backports release that corresponds to the Laird software release that you will be integrating. Once you have downloaded the correct release, extract the archive.

If your board support package and build system does not have CRDA available as a build target, then edit the .config file and find the following:

```
# CPTCFG_CFG80211_INTERNAL_REGDB is not set
```

Change it to the following:

```
CPTCFG_CFG80211_INTERNAL_REGDB=y
```

Find the db.txt file from the wireless-regdb 2013.02.13 package referenced in the Supporting Libraries and Programs section. Copy this **db.txt** over the placeholder file in **/net/wireless/** in the Backports release.

On your host build system, complete the following steps:

```
set -a
CROSS_COMPILE=${CROSS_COMPILE}
ARCH=${TARGET_CPU}
KLIB_BUILD=${DEV_PATH}/${LINUX_DIR}
KLIB=${TARGET_ROOTFS_ON_HOST}
set +a
make oldconfig
make
make install
```

After running make oldconfig, verify the following config values are set in the Backports .config:

```
CPTCFG_WIRELESS=y
...
CPTCFG_CFG80211=m
...
CPTCFG_WLAN=y
...
CPTCFG_ATH_CARDS=m
CPTCFG_ATH_DEBUG=y
CPTCFG_ATH6KL=m
CPTCFG_ATH6KL_SDIO=m
CPTCFG_ATH6KL_DEBUG=y
```

Note the following:

- `${CROSS_COMPILE}` – The location of your cross compiler.
- `${TARGET_CPU}` – Your target CPU architecture.
- `${DEV_PATH}/${LINUX_DIR}` – The location of your already-configured kernel build directory.
- `${TARGET_ROOTFS_ON_HOST}` – The location of your rootfs to install the modules into and fix kernel module dependency files.

The following files are generated by this process:

```
ath6kl_core.ko --- at ./drivers/net/wireless/ath/ath6kl
ath6kl_sdio.ko --- at ./drivers/net/wireless/ath/ath6kl
cfg80211.ko ----- at ./net/wireless/cfg80211.ko
compat.ko ----- at ./compat/compat.ko
```

If the `make install` command is not working then these files must be added to your device's root filesystem in **/lib/modules/<kernel name>/kernel/<each module path above>/<module>.ko** .

If you encounter dependency issues when inserting the ath6kl driver modules, run **depmod -a** to regenerate the kernel's module dependency file on a running device with the modules installed.

The following is the correct order of module loading:

1. compat.ko
2. cfg80211.ko
3. ath6kl_core.ko
4. ath6kl_sdio.ko

We recommend that ath6kl_core is loaded with the following module parameters:

```
recovery_enable=1 heart_beat_poll=1000
```

These modules parameters enable automatic firmware recovery on a radio firmware crash and heartbeat detection for recovering unresponsive radio firmware.

Laird-Provided Binary Files

If this is a new integration and you have not already been provided the following files, please contact ews.support@lairdtech.com.

If you already have contacted Laird support and have access to our download site, please visit: <http://www.lairdtech.com/products/msd45n>. Download the appropriate version of the Laird software release for Linux and extract the archive file. You will also need to extract the rootfs.tar archive file contained within the release archive file.

▪ Laird firmware files

These can be found under `/lib/firmware/ath6k/AR6003/hw2.1.1/` after extracting the rootfs.tar. These files should be placed by your build system into `/lib/firmware/ath6k/AR6003/hw2.1.1/` on your root file system.

- **bdata.bin** – This should be a symbolic link to the real `calData_XXXXXXXXXX_xxxx_CCA.bin`
- **calData_XXXXXXXXXX_xxxx_CCA.bin** – This is the board data file which includes unique settings and calibration for the module
- **fw-4.bin** – This should be a symbolic link to the real `fw_v3.x.x.x.bin`
- **fw_v3.x.x.x.bin** – This is the Laird enhanced radio firmware for normal runtime.
- **aththcmd_ram.bin** – This should be a symbolic link to the real `aththcmd_ram_v3.x.x.x.bin` (optional)
- **aththcmd_ram_v3.x.x.x.bin** – This is the firmware to use when wanting to run **aththtestcmd** for regulatory testing. (optional)

▪ Laird executable files

These can be found under `/usr/bin/` after extracting the rootfs.tar. These files should be placed by your build system into `/usr/bin/` on your root file system.

- **Sdcsupp** – This is the Laird supplicant. It provides enhanced authentication and controls the wireless connection from userspace. It should be started after the ath6kl driver has been loaded during system initialization. Laird recommends the following parameters when starting sdcsupp:

```
sdcsupp -iwlan0 -Dnl80211 -s
```

The `-i` specifies the wireless interface, `-D` enables using nl80211 to communicate with the kernel, and `-s` allows logging to syslog

- **sdc_cli** – This is Laird's CLI to control wireless operation. For more information, see the Laird SDC_CLI user manual.
- **dhcp_injector** – This is for injecting DHCP events into Laird SDK Events. See Laird's SDK documentation for further information. (optional)
- **aththtestcmd** – This is the Atheros test command program for regulatory and manufacturer testing. This should not be provided in normal production release and should only be used for testing (optional)

▪ Laird libraries files

These can be found under `/usr/lib/` after extracting the `rootfs.tar`. These files should be placed by your build system into `/usr/lib/` on your root file system.

- `libsdc_sdk.so.1` – this is a symbolic link to the current version of Laird’s SDK.
- `libsdc_sdk.so.1.0` – this is the Laird WiFi SDK and is required for the Laird CLI and supplicant to work. This SDK can also be used to allow your applications to interface with the Laird wireless package in C or C++. See the Laird SDK documentation for more information.

OPENSSL AND FIPS (NOT CURRENTLY SUPPORTED)

OpenSSL is required for the supplicant to work. The supplicant is the program that handles the various encryption protocols for WiFi. OpenSSL is the encryption library that our supplicant uses.

Building OpenSSL

As of this writing, we recommend OpenSSL version 1.0.1h. Earlier versions have known security bugs. This version requires a minor patch as shown below:

```
diff -purN openssl-1.0.1h.orig/ssl/s3_clnt.c openssl-1.0.1h/ssl/s3_clnt.c
--- openssl-1.0.1h.orig/ssl/s3_clnt.c      2014-06-05 02:44:33.000000000 -0700
+++ openssl-1.0.1h/ssl/s3_clnt.c          2014-06-08 10:19:21.643271429 -0700
@@ -901,6 +901,7 @@ int ssl3_get_server_hello(SSL *s)
     {
         s->session->cipher = pref_cipher ?
             pref_cipher : ssl_get_cipher_by_char(s, p+j);
+        s->s3->flags |= SSL3_FLAGS_CCS_OK;
     }
 }
#endif /* OPENSSL_NO_TLSEXT */
```

This patch can be provided by Laird in electronic form.

The proper flags for configuring OpenSSL are the following:

```
./Configure linux-armv4 --prefix=/usr --openssldir=/etc/ssl --libdir=/lib
threads shared no-idea no-rc5 enable-camellia enable-mdc2 enable-tlsext
zlib-dynamic fips
```

Include the last option, `fips` only if also building and using the FIPS component.

Building FIPS

The FIPS component is optional. It requires kernel support in the form of a special Laird-FIPS driver. It is only useful in the context of selling to the United States.

OpenSSL FIPS is a wrapper that needs to be built before the OpenSSL package. It may only be acquired through a method that satisfies the “trusted path” requirement. The one method known to satisfy this requirement is to obtain the source code distribution from the vendor of record (OSF) on physical media (CD). For instructions on requesting this CD see: <http://opensslfoundation.com/fips/verify.html>

We recommend that you become familiar with the OpenSSL FIPS library. Information is available here: <https://www.openssl.org/docs/fips/UserGuide-2.0.pdf>

OpenSSL FIPS doesn’t require any special flags for configuration.

Script

Laird provides a script that can be used, and modified as necessary for your environment, that will build both the FIPS and OpenSSL components in one shot. Please ask for the electronic copy of the script if you need it.

```
#####
installldir=$PWD/_install
sslfile=openssl-1.0.1h.tar.gz
fipsfile=openssl-fips-2.0.5.tar.gz
url=http://www.openssl.org/source
if [ ! -e $fipsfile ]
then
    wget $url/$fipsfile
fi
if [ ! -e $sslfile ]
then
    wget $url/$sslfile
fi

crossgcc=`which arm-sdc-linux-gnueabi-gcc`
cross=${crossgcc%-gcc}

export CROSS_COMPILE=$cross-
export HOSTCC=gcc
export INSTALL_PREFIX=$installldir

export MACHINE=armv5tej1
export RELEASE=3.8-laird1
export SYSTEM=Linux
export BUILD=sdcc

# check for zlib.h/zconf.h in toolchain
# note, for zlib-dynamic: need zlib.h/zconf.h in toolchain
sysroot=`$cross-gcc -print-sysroot`
if [ ! -r $sysroot/usr/include/zlib.h ]
then
    echo WARNING: missing zlib files in toolchain
    exit 1
fi

#####
# build openssl-fips
# note, refer to openssl security policy/guide for compliance
sslver=`ls -l openssl-fips-2.0*.tar.gz | sed -e 's/[.]tar[.]gz//g'`
tar -xf $sslver.tar.gz

cd $sslver
./config
make
make install
cd ..

#####
# copy incore to install directory for cross-compile build
installbin=$INSTALL_PREFIX/usr/local/ssl/fips-2.0/bin
cp $sslver/util/incore $installbin

#####
# build openssl-1.0xxx
sslver=`ls -l openssl-1.0*.tar.gz | sed -e 's/[.]tar[.]gz//g'`
tar -xf $sslver.tar.gz
# apply patch to fix EAP-FAST bug introduced in 1.0.1h
sed -n -e s/^[#]patch://gp $0 >$sslver.patch
cd $sslver
patch -p1 <./$sslver.patch
cd ..

export FIPSDIR=$installldir/usr/local/ssl/fips-2.0
export FIPS_SIG=$FIPSDIR/bin/incore

cd $sslver
./Configure linux-armv4 --prefix=/usr --openssldir=/etc/ssl --libdir=/lib threads shared no-
idea no-rc5 enable-camellia enable-mdc2 enable-tlsexp zlib-dynamic fips
```

```

make clean
make
make install
cd $PWD

#=====
ls -l $INSTALL_PREFIX/usr/bin
ls -l $INSTALL_PREFIX/usr/lib

#=====
# Patch to apply only to openssl-1.0.1h (not required for 1.0.1i or higher)
#patch:diff -purN openssl-1.0.1h.orig/ssl/s3_clnt.c openssl-1.0.1h/ssl/s3_clnt.c
#patch:--- openssl-1.0.1h.orig/ssl/s3_clnt.c 2014-06-05 02:44:33.000000000 -0700
#patch:+++ openssl-1.0.1h/ssl/s3_clnt.c 2014-06-08 10:19:21.643271429 -0700
#patch:@@ -901,6 +901,7 @@ int ssl3_get_server_hello(SSL *s)
#patch:
#patch:         {
#patch:             s->session->cipher = pref_cipher ?
#patch:                 pref_cipher : ssl_get_cipher_by_char(s, p+j);
#patch:+             s->s3->flags |= SSL3_FLAGS_CCS_OK;
#patch:         }
#patch:     }
#patch: #endif /* OPENSSL_NO_TLSEXT */

```

REVISION HISTORY

Revision	Date	Description	Initiated By
1.0	15 Oct 2014	Initial Release	Brian Wagner
1.1	24 Feb 2015	Changed backports information to Laird's version, changed references to Android specific file integration to Linux specific integration, added information on integrating regulatory database.	Dan Kephart